15｜音画同步：如何让声音和画面手拉手前进？

2021-12-27 李江

《攻克视频技术》 课程介绍 >

讲述：李江

时长 10:25 大小 9.55M

你好，我是李江。

在上节课中，我们讲述了音视频封装以及音视频数据是如何装到 FLV 和 MP4 文件里面的。这节课我们来讲讲播放这些文件的时候需要用到的一个非常重要的技术——音视频同步，也叫**音画同步**。

音视频同步是什么呢？它就是指在音视频数据播放的时候，播放的画面和声音是需要同步的，是能对得上的。相信你一定遇到过这种情况，就是看电视、电影或者直播的时候，人的口型和声音是对不上的，这样看起来会让人非常难受，这种问题就是音视频不同步导致的。因此做好音视频同步是非常重要的，当然也会有一定的难度，我们不妨先从一些基础知识讲起。

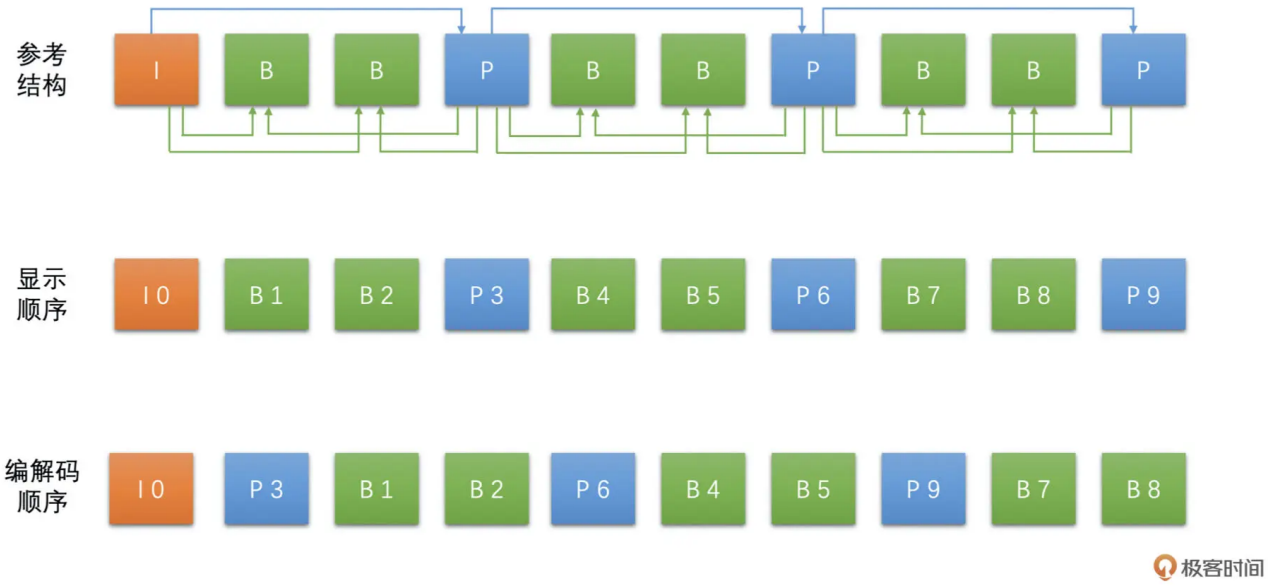
PTS 和 DTS

首先就是 PTS 和 DTS 这两个概念。其实，我们在讲音视频封装的时候已经提到过了。

PTS 表示的是视频帧的显示时间，DTS 表示的是视频帧的解码时间。对于同一帧来说，DTS 和 PTS 可能是不一样的。

为什么呢？主要的原因是 B 帧，因为 B 帧可以双向参考，可以参考后面的 P 帧，那么就需要将后面用作参考的 P 帧先编码或解码，然后才能进行 B 帧的编码和解码。所以就会导致一个现象，后面显示的帧需要先编码或解码，这样就有解码时间和显示时间不同的问题了。如果说没有 B 帧的话，只有 I 帧和 P 帧就不会有 PTS 和 DTS 不同的问题了。

具体如下图所示：



时间基

另外一个很重要的概念是时间基。这是面试中经常会问到的知识点，你一定要掌握。

时间基是什么呢？很简单，**它就是时间的单位。**比如说，编程的时候我们经常使用 ms（毫秒）这个时间单位，毫秒是 1/1000 秒，如果你用毫秒表示时间的话，时间基就是 1/1000。再比如说 RTP 的时间戳，它的单位是 1/90000 秒，也就是说 RTP 时间戳的时间基是 1/90000。意思是 RTP 的时间戳每增加 1，就是指时间增加了 1/90000 秒。

而对于 FLV 封装，时间基是 1/1000，意思是 FLV 里面的 DTS 和 PTS 的单位都是 ms。MP4 的话，时间基就是在 box 中的 time_scale，是需要从 box 中读取解析出来的，不是固定的，具体可以参考第 14 讲。这就是时间基的概念。

音视频同步的类型

好，了解了基础知识以后，我们就可以开始学习音视频同步了。音视频同步主要的类型有三种：**视频同步到音频、音频同步到视频、音频和视频都做调整同步**。我们逐一看下。

首先，视频同步到音频是指音频按照自己的节奏播放，不需要调节。如果视频相对音频快了的话，就延长当前播放视频帧的时间，以此来减慢视频帧的播放速度。如果视频相对音频慢了的话，就加快视频帧的播放速度，甚至通过丢帧的方式来快速赶上音频。

这种方式是**最常用的音视频同步方式**，也是我们今天讲述的重点，后面我们就会以这种方式来深入探讨其原理。

其次，音频同步到视频是指视频按照自己的节奏播放，不需要调节。如果音频相对视频快了的话，就降低音频播放的速度，比如说重采样音频增加音频的采样点，延长音频的播放时间。如果音频相对视频慢了，就加快音频的播放速度，比如说重采样音频数据减少音频的采样点，缩短音频的播放时间。

这里需要格外注意的是，当音频的播放速度发生变化，音调也会改变，所以我们需要做到变速不变调，这个你可以参考另外一个专栏《搞定音频技术》，里面有详细的讲解。

一般来说这种方式是不常用的，因为人耳的敏感度很高，相对于视频来说，音频的调整更容易被人耳发现。因此对音频做调节，要做好的话，难度要高于调节视频的速度，所以我们一般不太会使用这种同步方法。

最后一种是音频和视频都做调整，具体是指音频和视频都需要为音视频同步做出调整。比如说 WebRTC 里面的音视频同步就是音频和视频都做调整，如果前一次调节的是视频的话，下一次就调节音频，相互交替进行，**整体的思路还是跟前面两种方法差不多**。音频快了就将音频的速度调低一些或者将视频的速度调高一些，视频快了就将视频的速度调低一些或者将音频的速度调高一些。**这种一般在非 RTC 场景也不怎么使用。**

视频同步到音频

那么接下来我们就深入学习一下最常用的音视频同步，即视频同步到音频，它是怎么工作的。这里我们**参考 FFplay 的代码实现来讲解其原理**。

首先，我们使用的时间戳是 PTS，因为播放视频的时间我们应该使用显示时间。而且我们需要先通过时间基将对应的时间戳转换到常用的时间单位，一般是秒或者毫秒。

然后，我们有一个视频时钟和一个音频时钟来记录当前视频播放到的 PTS 和音频播放到的 PTS。注意这里的 PTS 还不是实际视频帧的 PTS 或者音频帧的 PTS，稍微有点区别。

区别是什么呢？比如说一帧视频的 PTS 的 100s，这一帧视频已经在渲染到屏幕上了，并且播放了 0.02s 的时间，那么当前的视频时钟是 100.02s。也就是说视频时钟和音频时钟不仅仅需要考虑当前正在播放的帧的 PTS，还要考虑当前正在播放的这一帧播放了多长时间，这个值才是最准确的时钟。

而视频时钟和音频时钟的差值就是不同步的时间差。这个时间差我们记为 diff，表示了当前音频和视频的不同步程度。**我们需要做的就是尽量调节来减小这个时间差的绝对值。**

那怎么调节呢？我们知道，我们可以通过计算得到当前正在播放的视频帧理论上应该播放多长时间（不考虑音视频同步的话）。计算方法就是用还没有播放但是紧接着要播放的帧的 PTS 减去正在播放的帧的 PTS，我们记为 last_duration。

如果说当前视频时钟相比音频时钟要大，也就是 diff 大于 0，说明视频快了。这个时候我们就可以延长正在播放的视频帧的播放时间，也就是增加 last_duration 的值，是不是视频的播放画面就会慢下来了？因为后面的待播放帧需要等更长的时间才会播放，而音频的播放速度不变，是不是就相当于待播放的视频帧在等音频了？

反之，如果说当前的视频时钟相比音频时钟要小，也就是 diff 小于 0，说明视频慢了。这个时候我们就缩短正在播放的视频帧的播放时间，也就是减小 last_duration 的值，是不是视频的播放画面就会加快速度渲染，就相当于待播放的视频帧在加快脚步赶上前面的音频了？

这里略有点绕，你可以停下来理一理，总之还是很好理解的。

那具体到底对 last_duration 加多少或者减多少呢？我们来看看 FFplay 的代码是怎么做的。

复制代码

```
/* called to display each frame */
static void video_refresh(void *opaque, double *remaining_time)
{
    ......
    if (is->video_st) {
retry:
        if (frame_queue_nb_remaining(&is->pictq) == 0) {
            // nothing to do, no picture to display in the queue
        } else {
            double last_duration, duration, delay;
            Frame *vp, *lastvp;
            /* dequeue the picture */
            lastvp = frame_queue_peek_last(&is->pictq); // lastvp是指当前正在播放
            vp = frame_queue_peek(&is->pictq); // vp是指接下来紧接着要播放的视频帧
            if (vp->serial != is->videoq.serial) {
                frame_queue_next(&is->pictq);
                goto retry;
            }
            if (lastvp->serial != vp->serial)
                is->frame_timer = av_gettime_relative() / 1000000.0;
            if (is->paused)
                goto display;
            /* compute nominal last_duration */
            // last_duration是lastvp也就是当前正在播放的视频帧的理论应该播放的时间，
            // last_duration = vp->pts - lastvp->pts。
            last_duration = vp_duration(is, lastvp, vp);

            // compute_target_delay根据视频和音频的不同步情况，调整当前正在播放的视频帧
            // 得到实际应该播放的时间delay。
            // 这个函数是音视频同步的重点。
            delay = compute_target_delay(last_duration, is);
            time= av_gettime_relative()/1000000.0;

            // is->frame_timer是当前正在播放视频帧应该开始播放的时间，
            // is->frame_timer + delay是当前正在播放视频帧经过音视频同步之后应该结束播
            // 如果当前时间time还没有到当前播放视频帧的结束时间的话，继续播放当前帧，并计算
            if (time < is->frame_timer + delay) {
                *remaining_time = FFMIN(is->frame_timer + delay - time, *remai
                goto display;
            }

            // 如果当前正在播放的视频帧的播放时间已经足够了，那就播放下一帧，并更新is->fr
            is->frame_timer += delay;
            if (delay > 0 && time - is->frame_timer > AV_SYNC_THRESHOLD_MAX)
                is->frame_timer = time;
            SDL_LockMutex(is->pictq.mutex);
```

```
47              if (!isnan(vp->pts))
48                  // 用当前视频帧的pts更新视频时钟
49                  update_video_pts(is, vp->pts, vp->pos, vp->serial);
50              SDL_UnlockMutex(is->pictq.mutex);
51              if (frame_queue_nb_remaining(&is->pictq) > 1) {
52                  Frame *nextvp = frame_queue_peek_next(&is->pictq);
53                  // duration是当前要播放帧的理论播放时间
54                  duration = vp_duration(is, vp, nextvp);
55                  // 如果视频时钟落后音频时钟太多，视频帧队列里面待播放的帧的播放结束时间已经
56                  if(!is->step && (framedrop>0 || (framedrop && get_master_sync_
57                      is->frame_drops_late++;
58                      frame_queue_next(&is->pictq);
59                      goto retry;
60                  }
61              }
62              ......
63              frame_queue_next(&is->pictq);
64              is->force_refresh = 1;
65              if (is->step && !is->paused)
66                  stream_toggle_pause(is);
67          }
68  display:
69          /* display picture */
70          if (!display_disable && is->force_refresh && is->show_mode == SHOW_MOD
71              video_display(is);
72      }
73      ......
74  }
75
```

我们再来看看最重要的函数 compute_target_delay 具体是怎么实现的。

复制代码

```
1  static double compute_target_delay(double delay, VideoState *is)
2  {
3      double sync_threshold, diff = 0;
4      /* update delay to follow master synchronisation source */
5      if (get_master_sync_type(is) != AV_SYNC_VIDEO_MASTER) {
6          /* if video is slave, we try to correct big delays by duplicating or d
7          // get_clock(&is->vidclk)是获取到当前的视频时钟，视频时钟 = 当前正在播放帧的pt
8          // get_master_clock(is)是获取到当前的音频时钟（在视频同步到音频方法的时候），
9         // 音频时钟 = 当前正在播放音频帧的播放结束时间 - 还未播放完的音频时长。
10        // diff等于视频时钟相比音频时钟的差值；
11        // diff > 0 表示视频快了；
12        // diff < 0 表示视频慢了。
13         diff = get_clock(&is->vidclk) - get_master_clock(is);
14         /* skip or repeat frame. We take into account the delay to compute the
15         // delay就是last_duration，也就是当前播放帧理论应该播放的时长。
16         // sync_threshold是视频时钟和音频时钟不同步的阈值，就取为delay也就是last_durat
```

```
        // 如果-sync_threshold < diff < sync_threshold的话就不需要调整last_durati
        // AV_SYNC_THRESHOLD_MIN是0.04秒，也就是40ms，
        // AV_SYNC_THRESHOLD_MAX是0.1秒，也就是100ms，也就是说音视频同步中，最大不同步
        sync_threshold = FFMAX(AV_SYNC_THRESHOLD_MIN, FFMIN(AV_SYNC_THRESHOLD_
        if (!isnan(diff) && fabs(diff) < is->max_frame_duration) {

            // 如果视频时钟比音频时钟慢了的时间超过了sync_threshold，则将delay（也就是l
            if (diff <= -sync_threshold)
                delay = FFMAX(0, delay + diff);

            // 如果视频时钟比音频时钟快了的时间超过了sync_threshold，并且delay（也就是l
            // 大于0.1秒（AV_SYNC_FRAMEDUP_THRESHOLD）的话，
            // 我们就直接将delay（也就是last_duration）增加一个diff，减慢视频的速度。
            else if (diff >= sync_threshold && delay > AV_SYNC_FRAMEDUP_THRESH
                delay = delay + diff;

            // 如果视频时钟比音频时钟快了的时间超过了sync_threshold，并且delay（也就是l
            // 我们就将delay（也就是last_duration）增加一倍，减慢视频的速度。
            // 这里和前一个条件处理的不同就在于delay（也就是last_duration）是不是大于AV
            // 上面不直接将delay翻倍应该是delay太大，大于了0.1秒了，超过了不同步阈值的最
            // 而这个条件里面delay翻倍而直接不增加diff的原因应该是一般帧率大概在20fps左右
            // 增加一倍也不会太大，毕竟音视频同步本来就是动态同步。
            else if (diff >= sync_threshold)
                delay = 2 * delay;
        }
    }
    av_log(NULL, AV_LOG_TRACE, "video: delay=%0.3f A-V=%f\n",
            delay, -diff);
    return delay;
}
```

结合代码我们可以看出，音视频同步并不是完完全全同步的，而是**通过调整正在播放的视频帧的播放时间来尽量达到一个动态的同步状态**，这个状态里面的视频时钟和音频时钟并不是完全相等的，只是相差得比较少，人眼的敏感度看不出来而已。这就是音视频同步的原理。

总结

今天我们讲述了音视频同步的相关知识。音视频同步主要的任务就是使播放的声音和画面能够对齐同步，防止出现声音和画面对不上的问题。主要的类型有三种，分别是视频同步到音频、音频同步到视频、音频和视频都做调整同步。

视频同步到音频是指音频的播放速度不需要调节，只调节视频的播放速度。如果视频相对音频快了的话，就减慢视频的播放速度；如果视频相对音频慢了的话，就加快视频帧的播

放速度。这种方式是最常用的音视频同步方式。

音频同步到视频是指视频的播放速度不需要调节，只调节音频的播放速度。如果音频相对视频快了的话，就降低音频播放的速度；如果音频相对视频慢了的话，就加快音频的播放速度。但是需要注意的是，音频速度变化会导致音调改变，所以要保证变速不变调。可由于人耳的敏感度很高，音频的调整更容易被发现，因此这种同步方式难度很高，所以一般不建议你使用它。

音频和视频都做调整是指音频和视频都需要为音视频的同步做出调整。比如说 WebRTC 里面的音视频同步就是音频和视频都做调整。整体的思路跟前面两种差不多，音频快了就将音频的速度调低一些或者将视频的速度调高一些，视频快了就将视频的速度调低一些或者将音频的速度调高一些。

之后，我们对视频同步到音频这种方式做了深入讲解。我们主要是通过计算视频时钟和音频时间之间的差值 diff，来调节当前播放视频帧的播放时间 last_duration。如果 diff 大于 0，则加大 last_duration 的值，让视频速度慢下来，等等后面的音频；如果 diff 小于 0，则减小 last_duration 的值，让视频播放的速度快起来，赶上前面的音频。这就是音视频同步的原理。

思考题

这节课我们开放讨论，谈谈你在这门课程中的收获吧？或者你还有哪些不懂的知识点都可以说给我听听，如果有必要的话我们还可以做一些针对性的讲解。

不妨大胆直言，我们畅快交流，留言区见！

分享给需要的人，Ta订阅后你可得 20 元现金奖励

生成海报并分享

赞 0 提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 加餐 | 基于纯浏览器的视频会议方案探究

下一篇 结束语 | 信心比黄金更重要

精选留言 (5)

写留言

springXu
2021-12-27

收获最大的是那加餐篇。哈！那篇内容讲述了用现代浏览器提功的能力来实现音视频的方案。 十多年前浏览器播放音视频的方案主流是flash又或者是定义标签+插件的形式。
老师能不能来一篇课程全貌图。比如您开这课的思路，为什么需要这些内容呈现出来。
您在学习音视频中走过的路和遇上的坑？ 对于要做音视频工作的人，需要哪些必备的知识以及建议是什么？

展开

共 2 条评论 > 4

springXu
2021-12-27

我想表达的是这位极客时间老师的想法:
我觉得很多人缺少的不是更高深的知识。而是缺少一个成长路线的鸟瞰图。不知道自己在什么位置，也不知道终点在哪里。埋着头学下去，学了很多东西但却不知道用在哪里。

能有学习路线导航图。...

展开

1

tony
2021-12-30

为什么会出现多个时间戳单位？大家不能统一吗

作者回复: 因为不同场景的精度要求不一样吧。就像普通的时间单位也有us、ms、s一样的。

paradise
2021-12-29

视音频同步原理就是通过比较视频和音频的播放时间来去动态调整视频当前显示帧的播放时间，以达到齐头并进的目的。

晓龙

2021-12-27

之前一直尝试学习音视频技术，但每次都是学了半吊子，学了忘，忘了就算，而且也不深入，老师的这堂课是我第一次比较系统的学习音视频技术，隔壁的《搞定音频技术》也在学习，非常感谢老师的付出，也非常珍惜这次学习机会。

下面我列出几点，我还想知道的：...

展开